DOCUMENT RESUME

ED 378 238 TM 022 616

AUTHOR Reimann, Peter
TITLE Supporting Instance-Based Learning in Discovery Learning Environments.
PUB DATE Apr 94
NOTE 11p.; Paper presented at the Annual Meeting of the American Educational Research Association (New Orleans, LA, April 4-8, 1994).
PUB TYPE Reports - Evaluative/Feasibility (142) -- Speeches/Conference Papers (150)

EDRS PRICE MF01/PC01 Plus Postage.
DESCRIPTORS *Discovery Learning; *Experiential Learning; *Learning Strategies; *Problem Solving; *Simulation
IDENTIFIERS *Case Based Tutor; Case Method (Teaching Technique)

ABSTRACT

Recent research has demonstrated that knowledge about specific instances may be of more relevance to reasoning than has previously been assumed. Students can rely on principles they have learned, or they can recall something similar previously experienced, and base the new prediction on it in an instance-based approach. Instance-based (or case-based) problem solving is important in simulation environments. The CAse-BAsed Tutor (CABAT) is a tool that supports case-based learning in a simulation environment, specifically the computer-simulated laboratory environment DIBI. CABAT stores simulated experiments done by the student and uses the episodic knowledge when the student does a new experiment, reminding the student of the former solution. The interface for elaborated examples is described, and some early empirical results with a chess game that used the same approach are presented. Three figures illustrate the simulation approach. (Contains 17 references.) (SLD)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

Supporting Instance-Based Learning in Discovery Learning Environments[1]

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)
This document has been reproduced as received from the person or organization originating it
Minor changes have been made to improve reproduction quality
Points of view or opinions stated in this document do not necessarily represent official OERI position or policy

Peter Reimann
University of Freiburg
Department of Psychology
Niemenstr. 10
D-79085 Freiburg, Germany
reimann@psychologie.uni-freiburg.de

PERMISSION TO REPRODUCE THIS MATERIAL HAS BEEN GRANTED BY
PETER REIMANN
TO THE EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

ED 378 238

1.0 Instance-based reasoning and learning

Most exploratory learning environments and instructional simulations in particular are based on the principle of learning by induction: The student can generate for herself specific instances and is supposed to generalize over such observations or experiments. From this point of view, the pedagogical idea behind simulation environments is completely in line with the idea in concept acquisition and problem solving research, namely, that learning consists of discarding specific, superficial information in favor of general, abstract information. However, as Medin & Ross (1989) point out, equating intelligence with the use of abstract knowledge, discarding specific, superficial information may be a dangerous oversimplification. As recent research on concept formation (e.g, Brooks, 1987) and problem solving (e.g., Ross & Kennedy, 1990) illustrates, knowledge about specific instances may be of more relevance to reasoning than has been assumed so far both in psychology and pedagogy.

Let me illustrate the point by looking at a specific example. In the computer-simulated laboratory environment DIBI (Stumpf et al. 1988) students can acquire knowledge about principles governing elastic impact phenomena by designing experiments and predicting the outcomes. Focusing on the prediction step, we can phrase this as a small problem solving task: Given the current experimental configuration, predict its outcome. In order to solve this problem, students can follow two strategies: They can rely on their more or less general knowledge (beliefs, hypotheses) about principles in the domain and apply this knowledge to the current case. Alternatively, they may recall a similar experimental design encountered before and base the new prediction on the observations made in the former, similar experiment. Roughly put, they may work principle-based or instance-based.

Taking the instance-based view seriously is important for two reasons. For one, working instance- (or case-) based is a powerful means for problem solving: If one does not have the required general knowledge (or applying it would be too resource-consuming because of the problem's complexity), then relying on instances and reasoning analogously may help. Secondly, instance-based problem solving may lead to more general knowledge, by learning processes such as incremental generalization.The current problem and the remembered/reconstructed similar solution instance form the basis for the generalization process. These remindings,

1. Paper given at the Annual AERA Meeting, New Orleans, April 4-8, 1994; Division C- Symposium: Learning in Computer Simulation Environments: An analysis of discovery learning.

TM022616

BEST COPY AVAILABLE

especially in novices, are often based on superficial features of problems, i.e., features that are not essential for problem solving. This must not necessarily impede learning because superficial features are often correlated with structural ones, and superficial and structural features become discriminated by incremental generalization. Incremental generalization is a by-product of analogical problem solving, not an automatic learning process. When the student has recalled a previous problem solving episode or example, she may attempt to map from the previous problem to the new one and transfer the former solution to the new problem. This will almost always require to ignore certain differences between the remembered and the current problem. To the extent that the old solution can still be mapped and the problem be solved, the differences ignored point out superficial features that one can generalize over. This incremental generalization method will lead to conservative generalizations containing only the minimal amount of abstractions required to make the mapping go through (see Ross & Kennedy, 1990, for more details and experimental support).

We claim that simulation environments, in particular those for semantically rich domains, should not only support the generation of generalizations (hypotheses) but also the management of specific information. This requires mainly to support the student in recalling fitting instances and transferring the former instance to the current problem. I will illustrate this idea with CABAT, a tool that supports case-based learning in a simulation environment, and SeeChess which was developed to elicit case descriptions from studying worked-out examples about chess endgames.

2.0 Supporting Instance-based learning in exploratory learning environments

2.1 Automatic Remindings

A first possibility to guide learning processes in the context of simulation environments is to support reminding. That can be done, for example, by providing the student with automatically generated remindings of previous problem situations that are in certain aspects similar to the current one. Following this idea, Schult (1993) developed the system CABAT ("CAse-BAsed Tutor") as a component of DIBI, the above mentioned microworld learning environment for conducting experiments concerning elastic impacts. The student's task in this microworld is to find out about regularities in the domain by arranging experiments and by predicting and evaluating their outcome. CABAT stores all experiments done by the student and uses this episodic knowledge whenever the student arranges a new experiment. If an analogous experiment can be retrieved, the system reminds the student of this previous one and explains the particular kind of similarity. CABAT shows the student the retrieved experiment, his prediction (hypothesis) of the outcome, and the system's feedback (correct solution), so that the student can use this former solution to predict the outcome of the current experiment. Similarity is computed based on domain formulas and several general heuristics in two steps. First, the formulas are simplified to gain constraints that can be used to classify situations in the domain with respect to structural features. Then, if there are several experiments that are structurally similar to the current one, one of them is selected based on superficial features. Since this procedure defines similarity only syntactically, it is in principle domain-independent. The only requirement is that the domain knowledge can be described by formulas (cf. Fig. 1).

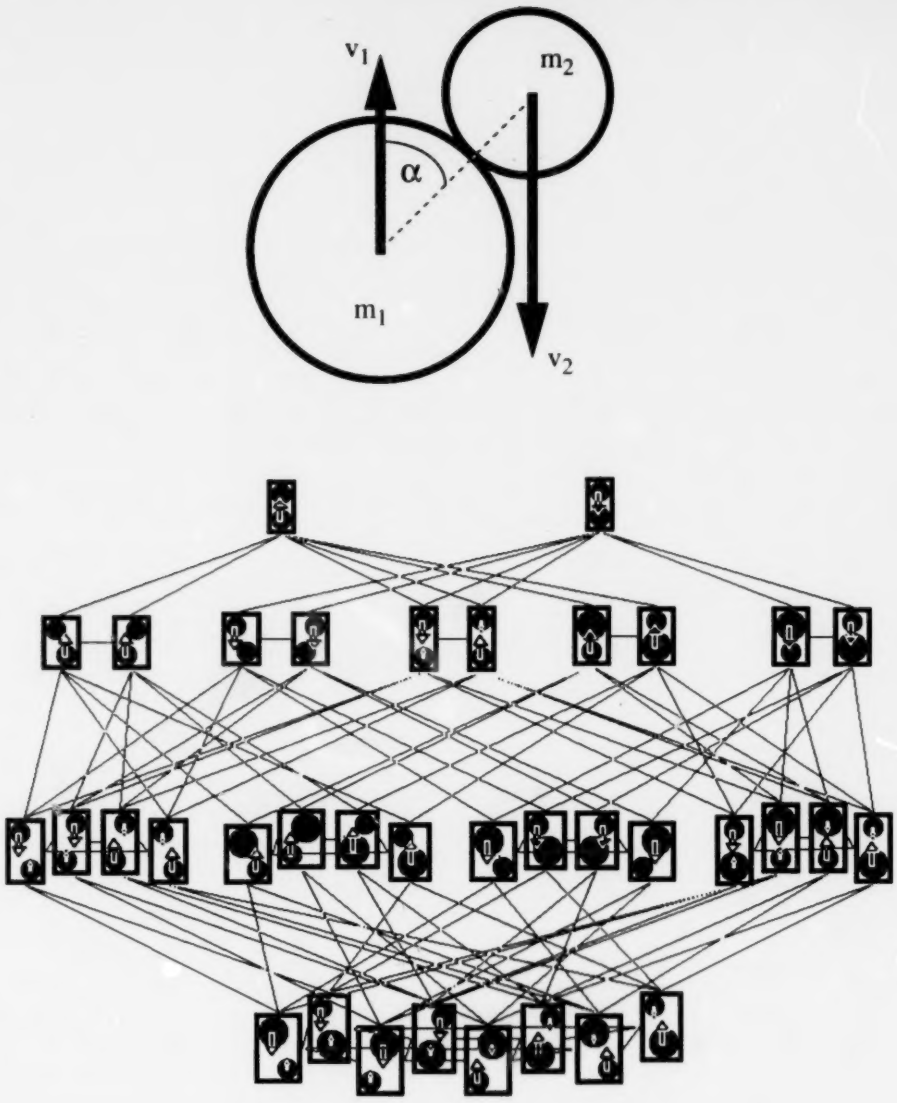

Figure 1.CABATs domain (a) and how it partitiones the space of experiments according to the similiarity between experiments (b)

CABAT solves the indexing problem for the student since it computes structurally important features automatically. Besides the fact that this is only possible in domains where structural features can be identified easily, this approach also assumes that the student will accept the cases retrieved by the computer as analogous, i.e., grasp the nature of the structural similarity. Not all students may be able to gain this sort of insight. Arguing from a pedagogical perspective, one may even claim that CABAT takes away an important task from the student, namely, to come up with descriptions of cases that identify superficial and structural features.

2.2 Annotating Learning Experiences

We (Beller, Reimann & Schult, 1994) have developed a tool to support instance-based reasoning in a domain where structural features are not easily identified and where it is up to the student to generate useful case descriptions. SeeChess allows a student to enter descriptions of worked out solutions to chess endgame problems in such a way into the computer that she can later one use this knowledge by means of a case-based reasoning module (see Figure 1). We use a hypertext tool to elicit from the student the necessary case descriptions[1] and case-base reasoning algorithms to provide a somewhat intelligent retrieval component[2]. The crucial point is that cases are produced by the student explicitly, with the understanding that they can be of use for later (case-based) problem solving. The instructional situation is one of *exploratory learning*: The relevant chess heuristics are not given to the learner, but he or she is supposed to infer them on the basis of analyzing worked-out examples and own problem solving experiences (i.e., playing the game). The computer tool in its current form (without a retrieval component) should help the learner to keep records of the relevant experiences and to structure this external „memory" of learning episodes in a way that makes it useful for later problem solving. In the next stage (not described here), when the program can actively support the student by providing remindings automatically (or on demand), the burden of searching memory will be lifted from the student and he or she can focus on adapting the former experience to the problem at hand.

It is not easy to provide an interface that lets students translate the externally provided example information and their own inferences into a case representation format. The problem is aggravated by the fact that students' conceptions about the features that are important to describe examples may change over time. For example, a learner may find out after having processed some examples that a property he didn't attend to so far is indeed important and should be included in all his example descriptions. We call this the *non-monotonicity problem*, an issue that plagues all knowledge acquisition methods, but is particularly frequent in novices' knowledge acquisition by novices.

What are the tasks a student is confronted with when describing an example to himself? These *elaboration tasks* (Chi, Bassok, Lewis, Reimann & Glaser, 1989; VanLehn, Jones & Chi, 1992) comprise:

- Selecting relevant given objects and their relevant properties
- Inferring the existence of objects and their properties
- Selecting relevant given relations
- Inferring additional relations
- Identifying the operators applied
- Inferring operators not explicitly specified
- Inferring conditions on operators
- Inferring effects of operators

---

1. By now realized
2. Under development

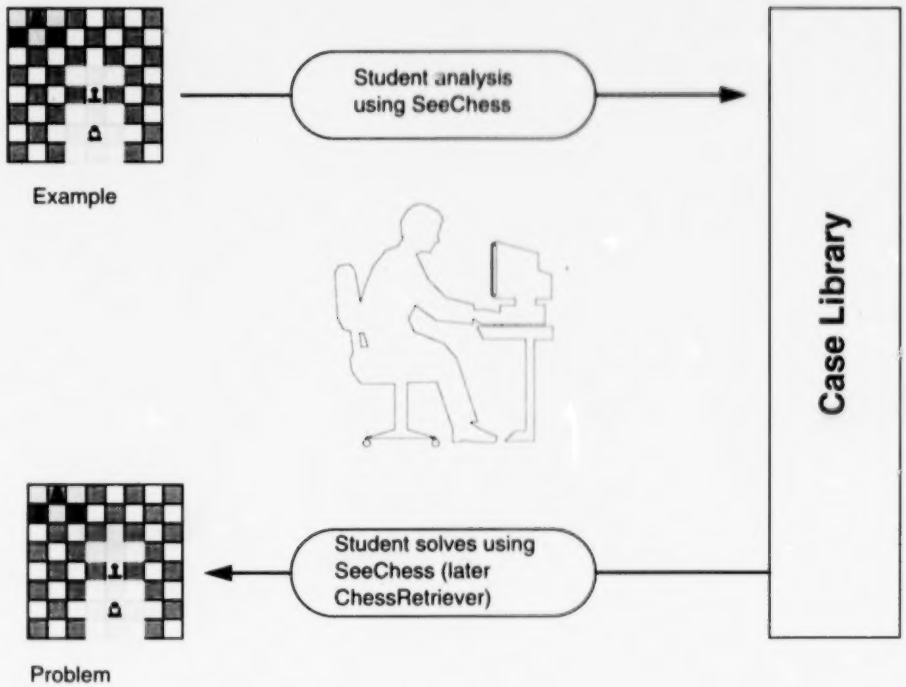

Figure 2.The Scenario

- Inferring goals
- Organizing operator applications (actions) belonging to a goal
- Inferring relations between goals

2.2.1 The Interface for Example Elaborations

In order to allow students to accomplish these tasks on a computer screen in a flexible manner without imposing too many constraints, tedious work or even programming on them, SeeChess provides them with a point-and-click interface, realized with Asymmetrix´ ToolBook under Windows. The students' task is to enter comments to the examples using the example analysis interface shown in Figure 2. In the upper portion of the screen, endgames examples are shown move by move on a board. In the lower part, the student can enter comments to the move that is currently shown on the board. Comments (i.e., elaborations) can be given either as free text or by using comment templates that cover frequently used, chess-specific components. The board is not only used to display example draws, but the student can also use it to perform further analysis; for instance, he or she can try out moves that are different from those in the example and watch the effects[1]. These variants can be annotated in the same manner as it it done with the example moves.

1. The elaboration tool is interfaced with a chess program (written in Lisp) that can play against the student. This program is also used to check the move suggestions of the student for correctness, i.e., whether they do not violate the chess rules. Illegal moves are rejected.

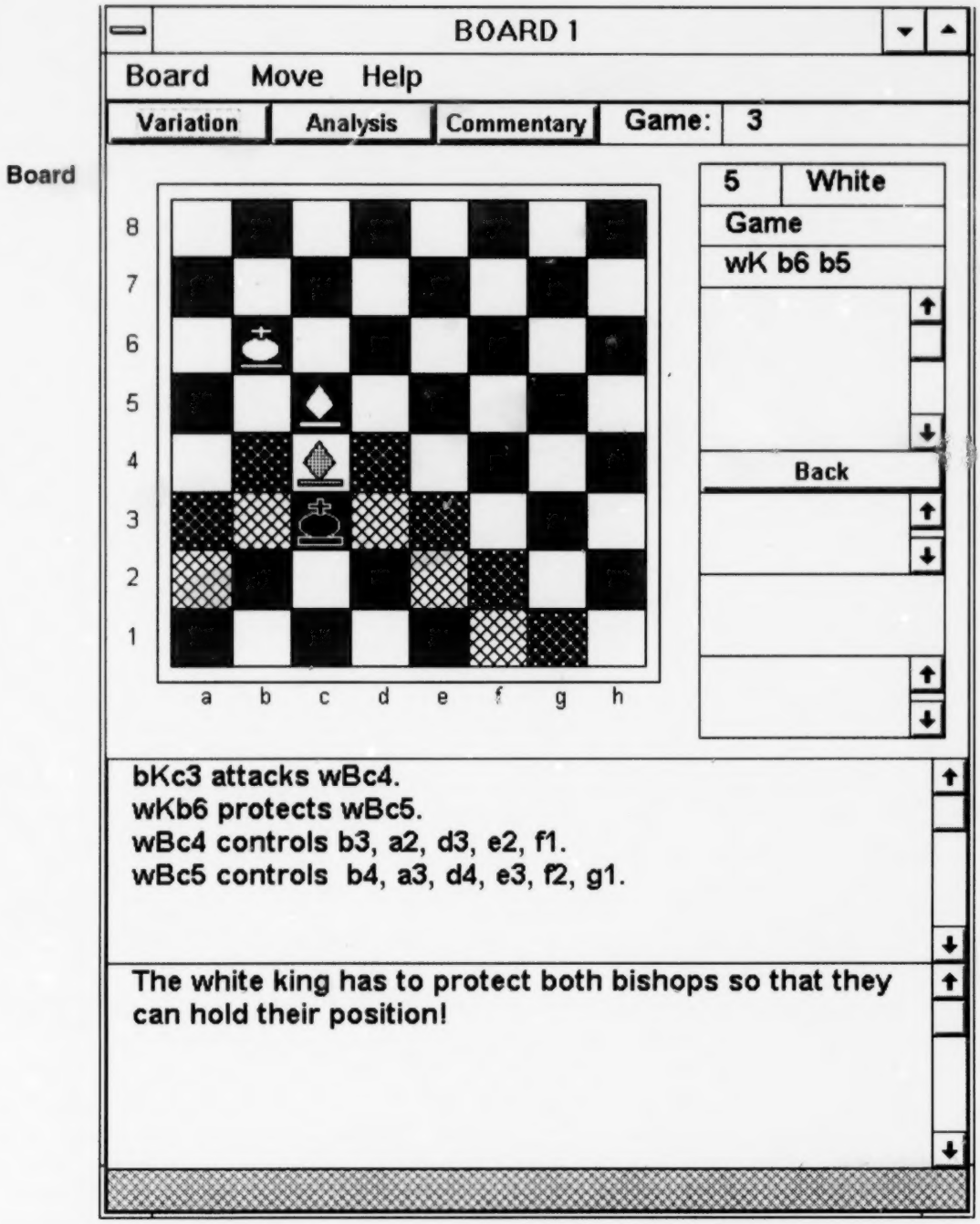

Figure 3.The analysis screen used to view examples and enter elaborations

2.2.2 Using elaborated Examples for Problem Solving

The elaborated examples are available to the learner when it comes to actually playing the game (against a computer chess program). In the current version, they are available exactly in the format that was used during the example elaboration phase, i.e., as a sequence of annotad moves. When working on chess problems (e.g., playing against a computer chess program) the student can page through the example games and look for situations that are similar to his or her current problem. (As mentioned before, in the next version of SeeChess, retrieval of similar example situations will be done in part automatically by a computerized module, called the ChessRetriever).

2.2.3 First Empirical Results

Empirical results are currently available on a episodic level, since data gathering was only recently completed. We ran a small study with three experimental conditions: Group 1 received an elaboration training before learning from examples; Group 2 did receive no such training; Group 3 did only learn by doing, i.e., did not see worked-out solutions. Group 3 received the (three) example solutions not as examples, but as training problems to work on. All three groups had to solve target problems, Group 1 and 2 after studying examples, Group 3 after working on training problems. All three groups used SeeChess to annotate examples and problem solving episodes, respectively. Subjects in all groups had basic chess knowledge, but did not play the game on a regular basis. Usually, they learned the basics during the youth and had not played the game since then.

The data analyzed sofar reveal that all three groups learn from the experience in the experiment, not only the two example-learning groups, but also the learning-by-doing group. However, it seems as if the two example-learning groups acquire more knowledge since these subjects are able to solve more problems that are dissimilar to the training problems. Sofar we cannot indentify a specific effect of the elaboration training.

Subjects show with varying frequency the following types of elaborations:

- Determining positive consequences of example moves (frequently)
- Determining the conditions for example moves (rarely);
- Determining positive and negative consequences for alternative moves (often).

What distinguishes good example elaborators from bad ones (as mentioned, this is as far as we can see by now not correlated with elaboration training)? The crucial difference seems to be that good elaborators are not satisfied with describing the examples moves in general terms (" king needs to be cornered in") but spend some effort to make such heuristics *operational*, that is, refine them down to a level that allows for move selection and execution (" king should not be able to move two squares toward the middle").

How did subjects use SeeChess? They all used it without feeling annoyed, and they used it after getting used to mousing fairly effectively (an individual session took three to four hours; one couldn't do it much faster with paper and pencil). In general, subjects annotated examples sparingly; in particular, they made little use of the opportunity to use color coding for representing features of piece configurations and they made usually only few free comments. This may be due to the fact that the endgame examples where of a basic type with only a few pieces on the board so that

the situation was managable relying only on mental capacities. Subjects did in the average use their annotated examples during problem solving, i.e., they worked in an analogical, case-based fashion.

3.0 Comparison with other Approaches to support learning in exploratory learning environments

Because of the cognitively challenging characteristics of exploratory learning environments, simulations in particular, a variety of approaches have been developed aimed at helping the learner to cope with these challenges. For instance,

- **Reification:** Visualizing concepts that are usually not visible and empowering the student to manipulate them. For instance, in ARK (Smith, 1986) theoretical concepts such as force and gravity are presented to the student in a way so that he or she can manipulate them directly. SeeChess incorporates a simple reification component: notions such as opposition and thread can be represented graphically on the board[1].

- **Data analysis tools:** In instructional simulations that are based on quantitative relations, students are often provided with tools to gather and analysis quantitative data (for instance, the spreadsheet tools in REFRACT (Reimann, 1992) and SMITHTOWN (Shute, Glaser & Raghavan, 1989)). The notion of "data" needs to be distinguished from the notion of a "case". A case has more of a macro-structure, is like a story, containing more or less complex sequences of situation - action - effect chains. One can learn a lot from a single case, whereas on can learn only from looking at many data.

- **Capturing theory:** Based on the problem solving theory of inductive learning in general and scientific discovery in particular (Simon & Lea, 1974; Klahr & Dunbar, 1988), several researchers have developed means to represent hypotheses and the relation between evidence and hypotheses in exploratory learning environments (Reimann, 1989; Shute et al. 1989; van Joolingen & de Jong, 1994). In SeeChess, it is not distinguished between evidence (i.e., examples) and hypotheses (e.g., heuristic rules) as two principally different classes of concepts. Rather, elaborations are seen as lying on a continuum ranging from inferences that go just a little bit beyond the information given up to abstractly formulated heuristics of how to play the game.

- **Coaching:** The most advanced but also most difficult to realize form of support attempts to infer the knowledge state of the specific learner from the interaction with the learning environment and to formulate advice for the student based on a comparison between the student model and a normative model of performance (e.g., SOPHIE (Burton, Brown & Bell, 1975), WEST (Burton & Brown, 1979)). Advice may also be given not on domain relations, but on "metacognitive" issues such as systematicity of discovery behavior (Burton et al., 1975; Shute et al., 1989). SeeChess will incorporate in its next incarnation a module that can providing students automatically with remindings about former situations that are

1. Since this is done using color coding, it is not displayed in Figure 2.

similar to the current one. This way, one can for instance remind students of similar mistakes they have made before. This is only an indirect form of coaching since it is up to the student to draw conclusions based on these remindings. The advantage of this approach however is that now explicit student model is required, but only a similarity measure. From a pedagogical point of view, it may make sense to give students a chance to diagnose their mistakes themselves (e.g., by comparing similar mistakes they made) before intervening with direct advice (as a coach would do).

References

Chi, M.T.H.; Bassok, M.; Lewis, M.; Reimann,P.; Glaser, R. (1989). Self-explanations: How students study and use examples in learning to solve problems. Cognitive Science, 13, 145-182.

Beller, S., Reimann, P., & Schult, T.J. (1993). SeeChess - a tool for annotating learning experiences in chess end games. Technical Report in preparation. Freiburg, FRG: University of Freiburg, Department of Psychology.

Brooks, L.R. (1987). Decentralized control of categorization: The role of prior processing episodes. In U. Neisser (Eds.), Concepts and conceptual development: Ecological and intellectual factors in categorization (pp. 141-174) London: Cambridge Univ. Press.

Brown, J.S., Burton, R..R., & Bell, A.G. (1975). SOPHIE: a step towards a reactive learning environment. International Journal of Man-Machine Studies, 7, 675-696.

Burton, R.R., & Brown, J.S. (1979). An investigation of computer coaching for informal learning activities. International Journal of Man-Machine Studies, 11, 5-24.

Joolingen, W. R. van, & Jong, T. de (1994). An extended duach search space model of discovery learning (Technical Report). Enschede: Univ. of Twente.

Klahr, D.; Dunbar, K. (1988). Dual space search during scientific reasoning. Cognitive Science, 12, 1-48.

Medin, D. L., & Ross, B.H. (1989). The specific character of abstract thought: Categorization, problem solving, and induction. In Ed. Robert J. Sternberg (Eds.), Advances in the psychology of human intelligence, Vol. 5. (pp. 189-223) Hillsdale: Lawrence Erlbaum , Hillsdale, NJ.

Reimann, P. (1989). Modeling scientific discovery learning processes with adaptive production systems. In D. Bierman, J. Breuker, J. Sandberb (Eds.), Artificial intelligence in education (pp. 218-227) Amsterdam: IOS.

Reimann, P. (1992). Modeling active, hypothesis-driven learning from worked-out examples. In E. De Corte, M. Linn, H. Mandl, L. Verschaffel (Eds.), Computer-based learning environments and problem solving (pp. 129-149) Berlin: Springer.

Ross, B.H., & Kennedy, P.T. (1990). Generalizing from the use of earlier examples in problem solving. Journal-of-Experimental-Psychology-Learning,-Memory,-and-Cognition, 161, 42-55.

Schult, Th. J. (1993). Tutorial remindings in a physics simulation environment. In P. Brna, S. Ohlsson, & H. Pain (Eds.), Proceedings of AI-ED 93, the World conference on Artificial Intelligence in Education (pp. 105-112) Charlottesville, VA: AACE.'

Shute, V., Glaser, R., & Raghavan, K. (1989). Inference and discovery in an exploratory laboratory. In P.L. Ackerman, R.J. Sternberg, & R. Glaser (Eds.), Learning and individual differences (pp.) New York: Freeman.

Simon, H.A.; Lea, G. (1974). Problem solving and rule induction: A unified view.. In L.W. Gregg (Eds.), Knowledge and cognition (pp.) Hilldale, NJ: Erlbaum.

Smith, R.B. (1986). The Alternative Reality Kit. An animated environment for creating interactive simulations. In IEEE (Eds.), IEEE Computer Society Workshop on Visual Languages, Dallas, TX, June. (pp.).

Stumpf, M., Brankat, S., Herderich, C., Newen, A., Opwis, K., Plötzner, R, Schult, T. , & Spada, H. (1988). The graphical user interface of DIBI, a microworld for collesion phenomena (Technical Report). Freiburg, FRG: University of Freiburg, Dept. of Psychology.

VanLehn, K.; Jones, R.M.; & Chi, M.T.H. (1992). A model of the self-explanation effect. Journal of the Learning Sciences, 2, 1-59.